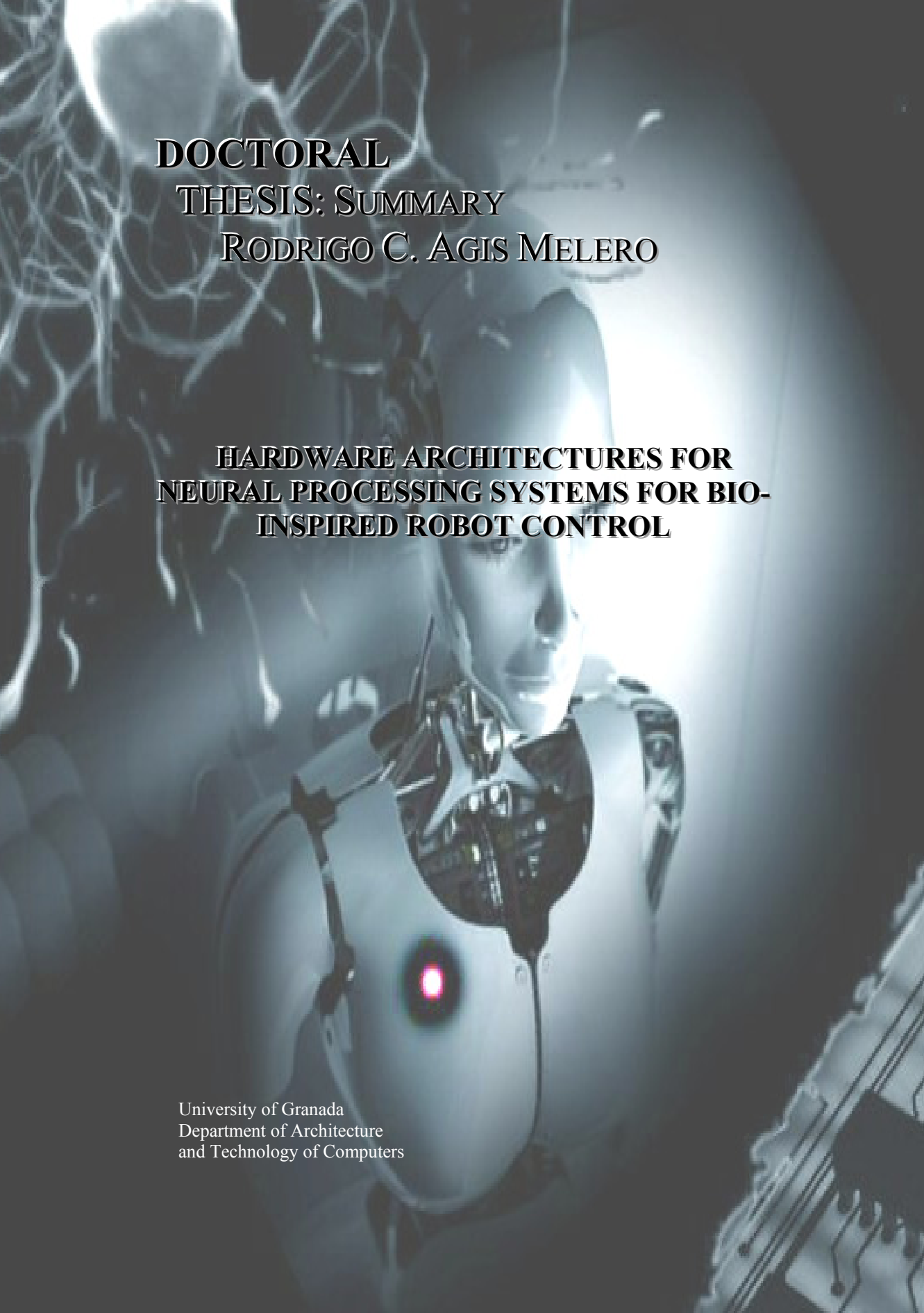

DOCTORAL THESIS: SUMMARY

RODRIGO C. AGIS MELERO

HARDWARE ARCHITECTURES FOR NEURAL PROCESSING SYSTEMS FOR BIO-INSPIRED ROBOT CONTROL

University of Granada
Department of Architecture
and Technology of Computers

Main index

1 Main Summary

This PhD Thesis is focussed in the study of spiking neural networks. In this framework the presented work presents different hardware architectures that are implemented in reconfigurable devices (FPGAs). Different approaches are proposed adopting *time-driven* or alternatively *event-driven* processing schemes.

The work presents alternative control approaches in the field of robotics and studies computing architectures for the simulation of massive spiking neural networks of millions of neurons processing sensorimotor information in real-time. These proposed approaches have been implemented in two hybrid Hardware/Software platforms with different levels of autonomy of the hardware (stand-alone and co-processing strategy) with respect to the software modules (in a PC as a host computer) that simulated in real-time these large scale networks.
In a second stage, this Thesis focuses on experiments with real-robots, as validation methodology of the control neural networks under study. The choice of working with real robots instead of simulated ones in motivated by the difficulty of describing in a realistic way the interaction with the real-world in a simulated framework. Therefore, the work here also adopts the "Embodiment concept" which stresses the necessity of having a physical body as learning mechanism for the knowledge emergence generation.

In this field, the Thesis describes two robotic platforms built and adapted for being controlled by spiking neural systems. The obtained results show that imitating in more or less detail the biology is feasible building neural circuits which represent valid alternatives to be considered for control of biomorphic robots with complex physical structures.

1.1 Introduction and motivation

Building artificial machines capable of performing tasks with "certain level of intelligence" (as humans do) has been one of the scientific goals in recent scientific history. Even having computers capable of performing millions of computations per second and diverse programming languages this goal remains as an open challenge. This is partly because current computing technology adopts mainly sequential processing strategies while natural systems (animals) use massive parallel computing in their central nervous systems for multi-sensorial perception tasks.

The interaction between artificial systems and the world is a challenge for classical problem solving methods in computation. Most of these interactions cannot be described as algorithms (nor can be simulated). Building an artificial model of the world or "environment" with a certain level of complexity is beyond current computers capabilities. Instead of building a global model of the environment, biological systems have developed mechanisms to dynamically adapt to the "experimented environment" and how to interact with it. Biological systems learn dynamically how to optimize their interactions with the environment through experience. This is done in the Central Nervous System (CNS) through neural circuits that receive, exchange, process and send information to their environment. This is done by means of spikes that encode all the signals being transferred or processed in the CNS. Spikes represent a good way of transmitting information with a low power cost and robust to noise. Building computing systems inspired in biological systems is not straightforward since we are trying to emulate a massive parallel computing system. For this purpose in this work we have chosen reconfigurable hardware (FPGA devices) that allows implementing parallel computing schemes. Even so, the number of computing elements that can be implemented with this technology is much reduced (far from the millions of units of biological systems). Nevertheless, we describe how to adopt opportunistic computing strategies to exploit efficiently certain characteristics of current technology (for instance using multiplexing techniques to take full advantage of high clock frequencies of current digital circuits).

1.2 Framework of research task

Before starting the work of this PhD different robotic platforms were developed that helped the author to gain expertise in this field. Furthermore we realised about the inherent difficulty of "creating new agents" and the challenge of devising new control schemes. This previous creative effort has highly motivated the work presented in this Thesis.

The Thesis has been done in the framework of two EU projects:

- **SpikeFORCE:** Real-Time Spiking Networks for Robot Control (FP5-IST-2001-35271). (01-05-2002 till 30-09-2005)

- **SENSOPAC:** Sensory motor structuring of perception and action for emerging cognition (FP6-IST-028056). (01-01-2006 till 30-12-2010)

The goal of SpikeFORCE was to develop models of spiking neural structures biologically plausible and study mechanisms for their utilization in robot control tasks. SENSOPAC represents a step beyond the control task and addresses the study of how, using biomorphic robots and bio-inspired control schemes we can evaluate efficient control and active sensing strategies for exploration tasks. In this project is studied how "cognitive notions" are structured by means of abstracted models in biological models, for instance in olivo-cerebellar structures and other neural subsystems.

For both projects the development of efficient spiking neural network simulation technologies to be used in real-time robot control tasks represents an issue of critical importance. Furthermore, the study of new computation schemes radically different to the conventional sequential processors based techniques remains as breaking through point in itself. The new "computing architectures" presented in this work can be considered as scalable architectures in which the input/output information is represented by means of neural

spikes. The processing architecture is a specific purpose processor with a single “instruction”: process a spike.

In this kind of architectures in possible to study the scalability issue and new parallelization techniques that become of specific interest provided the large number of computing resources currently available on single chips (reconfigurable hardware devices).

1.3 Objectives and organization

This Thesis presents different alternatives for designing massively hardware architectures. These architectures allow building neural circuits that emulate with a certain level of abstraction biological cell properties. Concretely, we will focus on simulating the cerebellum, due to its importance for coordination, movement control and motor learning.

Contrary to other classic artificial neural models, in which the internal dynamics of cells are just described by a predefined activation function (for instance a sigmoid function) the research work presented here focus on biological models of cells characterized by properties such as passive membrane potential decay, electrical coupling, gradual injection of charge, etc. In this way, it is possible replicating certain functionalities that seem to have a functional role in biological systems.

Many authors support the idea that the real potential of biological neural systems is facilitated by the network topologies and not by specific cell properties [23][1]. We think that both characteristics (cell temporal dynamics and network topologies) play complementary roles in the computing capabilities of biological nervous systems. This continuously motivates interdisciplinary works that try to build bridges between technology and neurobiology. Simulating biologically plausible cells and networks suggest new system properties that can open new research lines in the neurophysiologic field.

This thesis represents an effort also in building and dealing with real robots for evaluating and validating the neural computing engines that are described. This effort is partially motivated by the

"embodiment" concept [16][15] that suggest that is necessary a physical body to study mechanism of how knowledge emerges.
Another important aspect that is addressed in this work is the computing complexity of simulating massive neural systems (thousands to millions of neurons). To give an order of magnitude the human brain consists in approximately 20 thousand millions of neurons (with at least 1000 different types and 60 to the power of 14 synapses). Interestingly enough the cerebellum has 5 times this number of neurons (approximately one hundred thousand millions of neurons). The cerebellum is described as a set of computing elements with a well defined structure (climbing fibers, mossy fibers reaching granular cells, parallel fiber interconnecting Purkinge cells, etc). Diverse studies prove that only between a 5 to 10% of the cells are active at the same time (that brain uses sparse coding, especially in certain areas such as the Granular layer) which is also limited by the power consumption.

With the goal of simulating biological neural systems this work describes high performance computing engines for spiking neural networks and also how they deal with real robots.

For facilitating the easy reading of the Thesis, here we briefly describe a summary of the different chapters:

- **Chapter 1:** We briefly introduce the objectives and problems addressed in the different chapters of the work for the simulation of bio-inspired control systems.

- **Chapter 2:** We introduce different models of bio-inspired neurons (integrate and fire model, Spike Response Model, etc) and it briefly describes also certain Spike Time Dependent Plasticity (learning rules). This chapter tries to clearly distinguish conventional artificial neural networks and bio-inspired spiking neural networks with a certain level of biological plausibility.

- **Chapter 3:** We describe a co-processing computing architecture defined in reconfigurable hardware. We address in detail the characterization of the computing platform and its performance evaluation compared to sequential processing schemes. This

computing engine addresses time-driven simulations of spiking neural networks.

- **Chapter 4:** We describe in detail an event-driven computing engine with a pipelined datapath. We analyze its performance and the computing risks in its computing datapath and how the affect the system performance. One of the major contributions of this chapter is the parallel event selection architecture that enhances significantly the performance of the platform.

- **Chapter 5:** The robotics and its utilization to evaluate the hardware computing engines is one significant contribution of this Thesis described in this chapter. Here we describe different mechanic systems (robotic platforms) developed to study specific issues such as how to deal with robotic dynamics.

- **Chapter 6 and Chapter 7:** In these chapters we briefly describe the main contributions of the research work presented in this Thesis. We also indicate the scientific production of this research effort.

Note:
Since a more detailed description can be found in the Spanish version of the PhD we try to follow the same structure and refer to specific chapters along the summary.

The main contributions of this Thesis are described in Chapter 3 and Chapter 4.

In Chapter 3, a computing platform is described for simulating arbitrary networks of spiking neurons in real time. A hybrid computing scheme is adopted that uses both software and hardware components to manage the trade-off between flexibility and computational power; the neuron model is implemented in hardware and the network model and the learning are implemented in software. The incremental transition of the software components into hardware is supported. We focus on a Spike Response Model (SRM) for a neuron where the synapses are modelled as input-driven conductances. The temporal dynamics of the synaptic integration

process are modelled with a synaptic time constant that results in a gradual injection of charge. This type of model is computationally expensive and is not easily amenable to existing software-based event-driven approaches. As an alternative we have designed an efficient time-based computing architecture in hardware, where the different stages of the neuron model are processed in parallel. Further improvements occur by computing multiple neurons in parallel using multiple processing units. This design is tested using reconfigurable hardware and its scalability and performance evaluated. Our overall goal is to investigate biologically realistic models for the real-time control of robots operating within closed action-perception loops, and so we evaluate the performance of the system on simulating a model of the cerebellum where the emulation of the temporal dynamics of the synaptic integration process is important.

In Chapter 4 we have proposed a new simulation scheme for efficient simulation of spiking neural networks (SNN). Current SNN computing engines are still far away from simulating systems of millions of neurons efficiently. This chapter describes a computing scheme that takes full advantage of the massive parallel processing resources available at FPGA devices. The computing engine adopts an event-driven simulation scheme and an efficient next-event-to-go searching method to achieve high performance. We have designed a pipelined datapath in order to compute several events in parallel avoiding idle computing resources.

The system has a good performance and is able to compute approximately 2.5 million spikes per second although, how we will see, this performance will be decreased due to risks found in the pipeline data-path.

The whole computing machine is composed only by an FPGA device and five external memory SRAM chips. Therefore, the presented approach is of high interest for simulation experiments that require embedded simulation engines (for instance, in robotic experiments with autonomous agents).

Each chapter (3 and 4) focuses on the study of two different processing schemes (time driven and event-driven).
The first one uses a global clock for processing each neuron state, step by step. On the second one, the spikes are updated on chronological order avoiding the neuron processor to waste time between events. The global time clock is defined by the spike time label and therefore, the simulation engine is able to jump from one spike to the next one.
In this way, we can have a global vision about the best way for developing efficient hardware neuron processors (time and event-driven ones).

Since a more detailed description can be found in the Spanish version of the PhD we try to follow the same structure and refer to specific chapters along the summary.

2 Summary of chapter 3

A computing platform is described for simulating arbitrary networks of spiking neurons in real time. A hybrid computing scheme is adopted that uses both software and hardware components to manage the trade-off between flexibility and computational power; the neuron model is implemented in hardware and the network model and the learning are implemented in software. The incremental transition of the software components into hardware is supported. We focus on a Spike Response Model (SRM) for a neuron where the synapses are modelled as input-driven conductances. The temporal dynamics of the synaptic integration process are modelled with a synaptic time constant that results in a gradual injection of charge. This type of model is computationally expensive and is not easily amenable to existing software-based event-driven approaches. As an alternative we have designed an efficient time-based computing architecture in hardware, where the different stages of the neuron model are processed in parallel. Further improvements occur by computing multiple neurons in parallel using multiple processing units. This design is tested using reconfigurable hardware and its scalability and performance evaluated. Our overall goal is to investigate biologically realistic models for the real-time control of robots operating within closed action-perception loops, and so we evaluate the performance of the system on simulating a model of the cerebellum where the emulation of the temporal dynamics of the synaptic integration process is important.

2.1 Computing scheme

We have developed a hybrid software-hardware computing platform. The hardware component consists of an add-on board providing memory resources and an FPGA device that works as a reconfigurable neuro-processor. The software component runs on the host computer and the communication between the two is via the PCI bus. Currently the hardware component is restricted to computing the evolution of single-neuron state variables. The software component is responsible for maintaining the network

connectivity, for routing spikes between neurons, and for learning (defined as the plasticity of maximal synaptic efficacies). Communication between the hardware and software components is restricted to spike events; the software component sends packets of addressed pre-synaptic spikes to the hardware, and receives back packets of generated post-synaptic spikes. The neural states are stored in the co-processor board and do not need to be communicated at each epoch.

The communication between the software and hardware components is through packets of spike events. The software component generates pre-synaptic events and the hardware component generates post-synaptic events:

In the current implementation each neuron can have two types of synapses, one that is excitatory and one that is inhibitory. This will be extended in future implementations, where the pre-synaptic event structure will allow a model with a variety of distinct synaptic types such as AMPA, NMDA, GABAa or GABAb.

2.2 Neuron Model

The chosen neural model consists of terms to calculate the membrane potential together with a spike generation mechanism [24]. For a neuron with excitatory and inhibitory synapses, the subthreshold membrane potential Vx is given by:

$$V_x = V_x - \frac{1}{\tau_r}\left(V_x - V_{resting}\right) + G \cdot S_{exc} \cdot (U_{max} - V_x) - G \cdot S_{inh} \cdot \left(V_x - U_{min}\right)$$

Equation 1

This follows SRM in which the synapses are modelled as spike-driven conductance terms with synaptic efficacies that depend on the time elapsed since the received pre-synaptic spike (emulating the receptor mediated injection of charge).

The neuron is implemented using a processing unit (see Figure 1) with the following components:

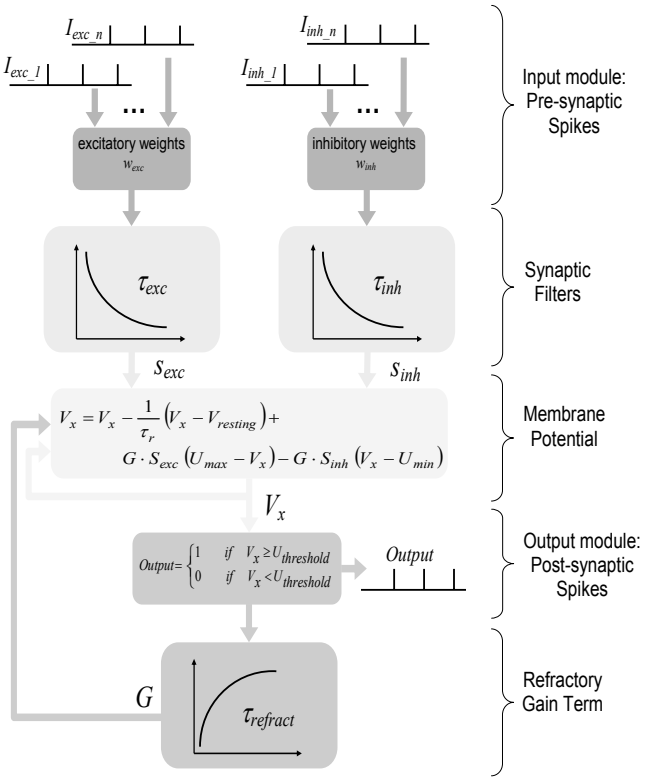

Figure 1: Processing unit scheme. The pre-synaptic events are summed up in software to reduce the communication costs. The computations indicated in this diagram can be pipelined as described in the next section.

2.3 Parallel computing strategies

A sequential implementation of the neuron model requires 17 time steps for all the computations. This is optimised by adopting a design strategy that efficiently exploits the parallel computing resources of FPGA devices.

Implementing algorithmic parallelism, or pipelining, is a frequently used technique in hardware design to reduce the number of time steps needed to perform complex operations.
In order to exploit the inherent parallelism of the FPGA devices, we have designed a pipelined computing structure with Nstages = 5 stages (S1 to S5) that iteratively updates the neural state variables.

These stages are:

1. **State fetch:** retrieves the neural state for each neuron from the neural state table (stored in the EMBs). This stage also takes the cell input, which can be an input spike from the pre-synaptic event table (stored in the embedded memory blocks, EMB) or zero in the absence of pre-synaptic events.

2. **Neural computation I:** calculate the synaptic gain terms using the IIR filters (Sexc and Sinh).

3. **Neural computation II:** calculate the membrane potential (Vx)

4. **Neural computation III:** calculate the resting and refractory components (G), and generate a post-synaptic spike if required.

5. **State write-back:** store the output spike and the neural state in the tables.

It is worthwhile to note that further neural features such as NMDA channels, firing threshold oscillations, etc; can be processed in this pipeline structure in specific stages.

Since all the stages are computed in parallel, we could include new neural features without degrading significantly the computation speed by processing them in extra pipelined stages (provided that the maximum number of cycles per stage is kept low).

2.4 Computational resources consumption

The computational load has been distributed into different pipelined stages to allow parallel processing along the datapath. The computational resources of each of these stages are summarized in Table 1 We have used the RC1000 board of Celoxica [3] as prototyping platform to evaluate the computation scheme. This is a PCI board with 4 banks of SRAM memory on board (2MB each) and a Xilinx device with two million gates (Virtex- 2000E) [26].

# PUs	# EMB	% EMB	# slices	% slices	# equivalent gates	#time steps
1	80	50	4581	23.86	1389247	6

Table 1: Computational resources of a complete pipelined processing unit. Total amount of resources used and as a percentage of the Xilinx device XCV2000-E. This device includes only general purpose processing circuitry (slices) and embedded memory resources (160 blocks of 4 Kbits each). This processing unit is able to compute 6 neurons in parallel using a pipeline processing scheme.

2.5 Computing Performance

In previous subsections we described the computing scheme. On this case the system adopts a hybrid architecture (hardware/software). The communication between hardware/software is made over the PCI-bus and adopts an epoch-driven computing strategy.

The time taken by the hardware to compute each epoch to epoch is given by:

$$t_{epoch} = \sum_{i=0}^{N_t-1} t^i_{TimeCycle} = \frac{1}{F_{clk}} \sum_{i=0}^{N_t-1} NTS^i_{TimeCycle}$$

Equation 2: $t^i_{TimeCycle}$ is the computing time of the time cycle i (i = 0... Nt –1). $NTS^i_{TimeCycle}$ is the number of time steps to compute time cycle *i*, and *Fclk* is the clock frequency.

On the other hand, the time taken by the processing units is fixed and depends on the number of neurons ($N_{neurons}$), the number of processing units (N_{PU}), the number of time steps of the longest stage of the pipeline structure (NTS_{lps}), and the latency ($L = N_{stages} \cdot NTS_{lps}$). The computing time of each epoch t_{epoch} can be given by Equation 3.

$$NTS^{i}_{TimeCycle} = \mathrm{MAX}\left(1+\left(N^{i}_{pre-synaptic}\cdot NTS_{Load-pre}\right),\ L+\left(\frac{N_{neurons}\cdot NTS_{lps}}{N_{PU}}\right)\right)$$

Equation 3

For instance choosing the following configuration: N_t = 20 cycles, $N_{neurons}$ = 1024 neurons, NTS_{lps} = 6 time steps and F_{clk} = 25 Mhz, then the time taken by the hardware using a single processing unit to compute one epoch is $t_{epoch} \approx 4.94$ ms.

A suitable integration step for biologically plausible neurons based on the chosen neuron model is 100μs. For a simulation to be in real time, an epoch of 20 time cycles must be computed in less than 2 ms.

To achieve this we use several computing processing units in parallel. In this way, the computing time decreases with the number of processing units (given for 1024 neurons, 20 time cycles per epoch, an integration time step of 100 μs, and variable numbers of pre-synaptic events). The results show that the design is scalable and the processing inherently parallel when the number of pre-synaptic events is not too high.

2.6 Real-time Cerebellar simulations

The performance of the system is dependent on the network connectivity and the load given as the number of pre-synaptic events per epoch. It is therefore important to test its performance within a biologically realistic context.
For this we simulate a model of the olivary-cerebellar system applied to a simple tracking task (visual smooth pursuit).

Movement in one dimension is modelled using two olivary-cerebellar microcircuits, each representing movement in opposite directions.

Full details of the model and its application to visual smooth pursuit are given in [9].

Of significance to this discussion are the following points:

1. The number of connections to each neuron varies greatly. The cerebellum does not represent a heterogeneous network of neurons, but contains both extremely low and extrememely high patterns of connectivity. Each granule cell receives less than ten inputs whilst each Purkinje cell receives hundreds of thousands.

2. The number of pre-synaptic events per epoch (and the computational load) is widely distributed in the simulations.The cells in the inferior olive fire at around 1Hz, those in the granular layer appear to be mostly silent with short bursts up to 100Hz, whilst the Purkinje cells may average around 80Hz.

3. The cerebellum is an example of a brain system that needs to be modelled in real time. The cerebellum is involved in the fine control and coordination of timed movement (amongst much else) but its exact role in the brain is unknown. The simplified and artificial version of the smooth pursuit problem used for this demonstration of the hardware can be easily simulated and is not a difficult task for the cerebellum to learn. Providing the realistic and sufficiently rich context that will be needed to elucidate cerebellar function implies being able to study cerebellar models operating in real time in the real world.

2.7 Discussion

We have described a hybrid hardware-software platform for the real time simulation of spiking neurons based on the Spike response neural model including a gradual injection of charge and synapses modelled as input-driven conductances.

The computation scheme is without any of the restrictions on the model imposed by current event-driven schemes.

The hardware co-processor is based on reconfigurable hardware (FPGA) and we have adopted a computing strategy that makes exhaustive use of the processing parallelism resources of the FPGA device.

Different hardware and software accelerators for simulating spiking neural networks simulations are difficult to compare since the time resolution of the integration, their objectives and the neuron models are all different. Compared to other specific computing platforms [7] [2] [14] [21] [11] our approach addresses the simulation of smaller scale networks but with a higher time resolution (100μs) because focus on real-time applications.

Our simulation platform has been designed for robotic applications and sensorimotor studies. Here is critical the capability of simulating different network topologies of small and medium size. Furthermore, the possibility of easily testing different networks for specific processing tasks makes FPGA technology the best choice for our approach.

The cell model includes the gradual injection of charge in the synaptic modules. This feature allows the study of synchronization processes. How synchronization arises from specific topology instead of intrinsic cell synchronisation mechanisms is an important topic.

FPGA technology has experienced great advances in recent years and is continuously evolving. We have described the circuits with a high level hardware description language [22], which facilitates the efficient management of the memory and computational resources. The design is modular and can be easily customized with different levels of parallelism.

The computing scheme is fully scalable, even though the analysis of the consumption of resources is based on a specific prototyping platform and a reduced number of neurons. There are two main factors that limit the scalability to very large scale simulations:

- The software/hardware communication bandwidth (that motivates the future use of PCI-X boards)

- The limited computing parallelism (that increases with more powerful FPGA devices, provided that the on-chip computing scheme is scalable).

The comparison of the proposed computing platform with event-driven simulation schemes is not easy, since the adopted neuron model implements mechanisms for the gradual injection necessary for defining synapses with different time constants. This feature is difficult to incorporate into an event-driven simulation schemes although some approaches are exploring this possibility with different restrictions [17].

3 Summary of chapter 4

3.1 Introduction

The simulation of spiking neural networks using standard integration methods on conventional computational architectures (single- or multiprocessor platforms) is inefficient. This has motivated the implementation of specific hardware platforms to perform neural integration.

The neuron model described on this contribution is the same as in Chapter 3, but the platform presented here adopts an event-driven scheme entirely simulated in hardware using parallel resources of FPGA devices. We implement a pipelined processing structure to further accelerating the simulator, but this requires the consideration of inter-spike dependency risks.

In this work, we present a specific purpose computing architecture to efficiently simulate spiking neural networks by adopting an event-driven scheme. The common approach is based on a queue of events ordered chronologically [20][13][8][18]. In this case, the goal is to reduce the number of accesses required for the correct insertion of a new spike in its correct position in a chronologically ordered lists. Contrary to this approach, we use a disordered event list, Our processing scheme searches for the next-event-to-go before each computing loop. For this purpose, we implement a parallel searching strategy that takes full advantage of the parallel processing resources available in FPGA devices.

3.2 Description of the Computing Scheme

The computing scheme is illustrated in Figure 2. The event list is stored on embedded memory resources in order to facilitate the insertion and searching processes. On the other hand, the neural state variables and the network topology are stored on external memory SRAM.

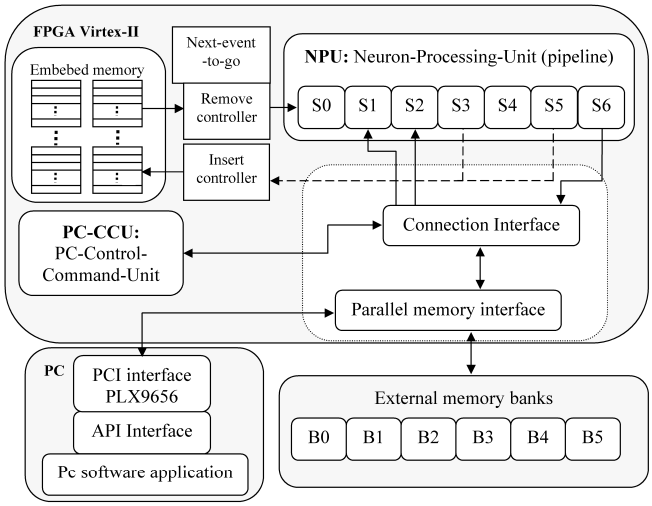

Figure 2: Computing Architecture Schematic

3.3 Next-Event Selection strategies

In order to take the "next even to go" we use a disordered event list. In this case, every time we need to extract an event, we search for the one with a minimum *time label*. We have implemented a parallel searching strategy taking full advantage of the parallel computing resources of the FPGA devices. It uses parallel comparator circuits of 4 and 8 elements each (see Figure 3). With this scheme we are able to manage event lists of up to 2^{14} elements consuming up to 69 clock cycles to take the next event to go.
Note that in order to pipeline this processing datapath we need to store in buffers (distributed memory) not only the time labels but also an index to identify the original spike (in the embedded memory block that is being processed).

Note that it is possible to use another selection algorithms such as Heap-Short, Quick-Short, etc., but these are inefficient when implemented in FPGA devices since they do not take advantage of parallel devices. In this chapter we have establish a comparative between Heap-Short algorithm and how despite the difference between the efficiency order, the parallel searching strategies becomes more efficient.

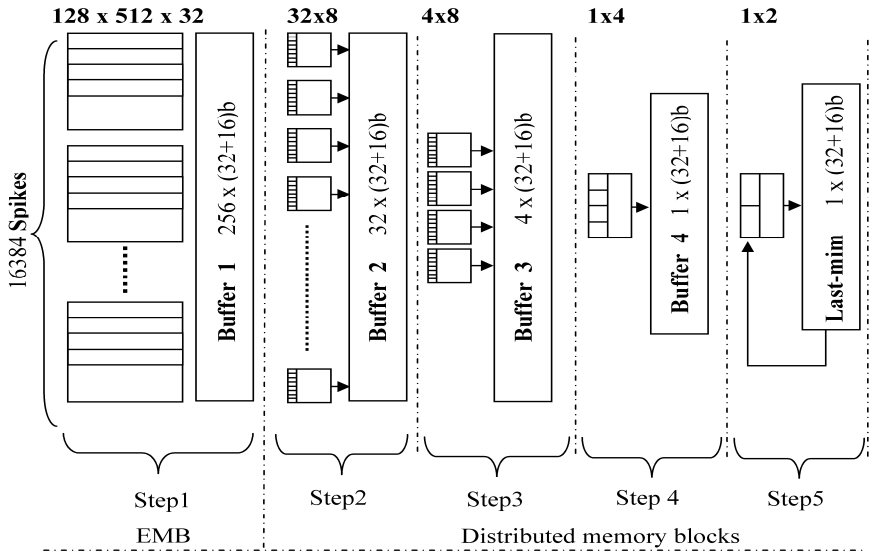

Figure 3: Parallel Searching Tree.

3.4 Store Memory Resources strategies:

On this chapter we have tested different Next-event to go selection strategies. Two factors become important to achieve high performance during the searching and retrieval of the *next-event-to-go*:

1. Maximum parallelism on the next time label searching.
2. Maximum parallelism on the *next-event-to-go* data fields reading.

There are different strategies that can be adopted as discussed in the chapter:

- Complete Events on External Memory Resources.
- Complete Events on Embedded Memory Blocks.
- Time Labels on Specific EMBs.
- Event Fields on Interleaved Embedded Memory.
- Some Event Fields stored on Specific Embedded Memory Blocks.

In this subsection a whole of strategies has been analyzed. We show a couple of measurements efficiency equations.

Finally we have adopted a "Complete Events on Embedded Memory Blocks" strategy for developing our neuron processor because it achieved a good trade-off between efficiency and hardware resource consumption.

3.5 Pipelined Event-processing Datapath

We have implemented a pipelined event-processing datapath consisting of the 9 stages:

SS: Shift stage (intermediate register with the partial results are updated).

S0: the *next-event-to-go* is searched (this is done through a parallel searching tree, as described in the previous section)

S1: Access to external memory is gained in order to retrieve the source neuron state variables and the connection characteristics:

- Access to external memory to recover the state source neuron.
- Recover state of the target neuron.
- Gets the connection between source and target neuron (topology).

S2: The spikes of the output pre-buffer are shifted (spikes thrown out)

S3: Spikes of the input pre-buffer are shifted (new input spikes produced during datapath processing). If there remains any spike of the output connection tree to be processed of this source neuron connection tree is inserted into the event list.

S4: Learning, the learning rule is processed.

S5: The target neuron state is updated (synaptic contribution processing)

S6: "*Axon Hillock"*. The Axon-Hillock is processed (spike firing decision)

S7: The connections states between source and target neuron are stored.

S8: The target neuron state and connection weight is stored.

Note: When a neuron fires, it produces multiple spikes that will reach different target neurons according to the network topology. In order to restrict the number of spike insertions (due to the neuron fan out), we consult the output connection tree in each computation cycle (ordered according to the synaptic delays) of a neuron that has fired, and we insert just the next-event according to the synaptic delay. This keeps the event list at a manageable size.

On this subchapter we estimate the performance of the system and degradation for a specific neural model due to datapath risks. This risks have been analyzed establishing management strategies.

We have defined the next management strategies:

1. **Synchronization strategies** (between stages)

2. **Risk resolutions strategies** (inter-block between different process that try to pick up share resources, blocks due to data dependency risk and give up the datapath).

3. **Load swinging strategies** (between the different datapath stages).

3.6 Neural Model

The described general architecture is valid for multiple neuron models. In fact, the neural state computation is a single processing stage that can be seen as a black box.

The only restriction is that the neural model allows the neural state variables to be updated discontinuously. Currently, we are using the proposed platform to test bio-inspired robotic control experiments.

We have proved the system with four different neural cells (Purkinge, Granuler, Golgi and Interneuron), following a cerebellar architecture. The general model includes (STPD) "*Spike Time Dependent Plasticity"* and passive decay terms.

3.7 Simulation Performance and Hardware Resources

It is difficult to compare the performance with other approaches, since each of them use different neural models. Currently, one of most efficient event-driven software versions [18] is able to compute up to 0.8 Mspikes/second using an AMD processor at 2.8 GHz. It is significant to note that, through the design of a specific purpose datapath working at a clock rate about 2 orders of magnitude lower than conventional computers; we are able to outperform in more than a factor of 2 the processing performance. Other simpler spiking neurons simulators are able to process higher rates [4] [5] but only including simplified neural models network topologies.

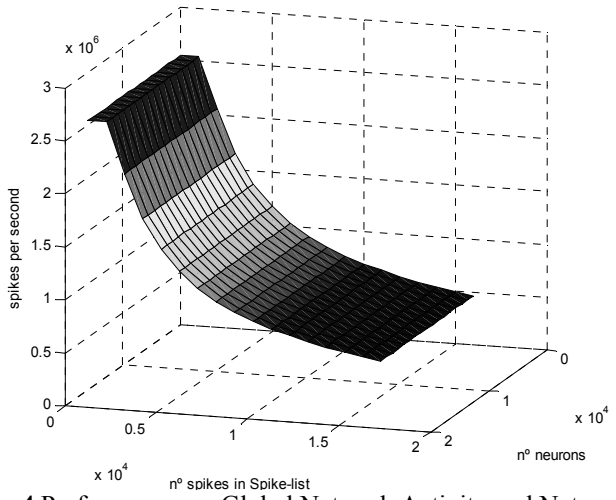

Figure 4 Performance vs. Global Network Activity and Network Size

It is also remarkable that the exploration of other neural models (even of a higher complexity) would not significantly degrade the system performance if the computation can be done in less than 27 independent steps or split in several pipelined processing stages.

Pipeline State	#System Gates	EMBs	# Clock Cycles
S0	4,823,495	64	10-69
S1	2450	-	9
S2	919	-	9
S3	1172	-	12
S4	1955	-	9
S5	1182	-	13
S6	1451	-	9

Table 2 Hardware Resources Consumption: Design Compiled on a Virtex II 6000 of Xilinx.

The data throughput (D_t) follows Ecuación 4, which is independent from the network size and includes a degradation term (A_{risks}) dependent on the inter-spike risks. This factor will not be significant in realistic networks in which spikes of output connection trees will be almost consecutively processed.

$$D_t = \frac{f_{clk}}{A_{risks} + \max\left[27,\ N_{clk\_cycles\_search}\right]}$$

Ecuación 4

The performance follows rigorously the characterization expression outlined above. The surface in Figure 4 has been done using a network topology (all-to-all connectivity with short synaptic delays). In this case, the inter-spike risks do not affect significantly the system performance. As can be seen, the performance does not depend on the network size, only on the global activity achieving a maximum performance of 2.5 millions spikes per second. The hardware resources consumption is summarized in Table 2.

3.8 Discussion

The main innovation of the presented approach is the efficient use of the parallel computing resources of FPGA devices for an event-driven processing scheme. We have adopted a strategy

that handles efficiently disordered event lists, which is a completely novel approach in the framework of event-driven spiking neural network simulation. We have used extensively parallel computing in the *next-event-to-go* searching structure that has been implemented with a finely pipelined searching tree.

The whole computing scheme is also implemented in a coarse pipelined datapath of 9 stages. Here, we need to handle inter-spike risks and this depends of the network characteristics. The performance degradation is not significant in realistic networks in which spikes of specific output connection trees will be processed almost consecutively.

Another important point is that, since the described computing platform is very general and can be easily adapted for different neural models, it becomes of interest in the framework of massive simulations and real-time experiments (for instance, in robotic experiments learning with sensory-motor integration schemes).
Finally two main aspects are considered i.e. accessing time and scalability: We can conclude that storing the events on embedded memory blocks maximizes the parallel access. Therefore, this represents the most powerful choice for an event list of a moderate size (several thousands of events), keeping in mind that the restricted number of embedded memory blocks limits its scalability.

4 Brief summary of chapter 5

The validation of spiking neural networks using standard simulators on conventional environments (non realistic environments) is incomplete. According to the "embodiment" concept, that suggest that it is necessary a physical body to study mechanism of how knowledge emerges, we have built and adapted two robots: A bio-inspired robots arm and a humanoid (see Figure 5 and Figure 6).

For both of them, a completed hardware platform description is made. With the first one, we use a standard PID controller and single experiments. We use genetic algorisms for to tuning the PID variables. The system is able to learn witch constants are the best for tuning the PID on specific trajectory learning the dynamic model.

This platform is bio-inspired and was developed trying to emulate a human-arm. It has no rigid joins (to allow taking advantage of the inertial components) and we can control directly through applied forces, while getting positions and the consumption of the of the motors system.

The robot arm is handled by FPGA platform that could be an interface with the PC or a Hardware neural simulator. With this platform we have fix the requirement of the real physic systems and we have a look on the field bio-robotics.

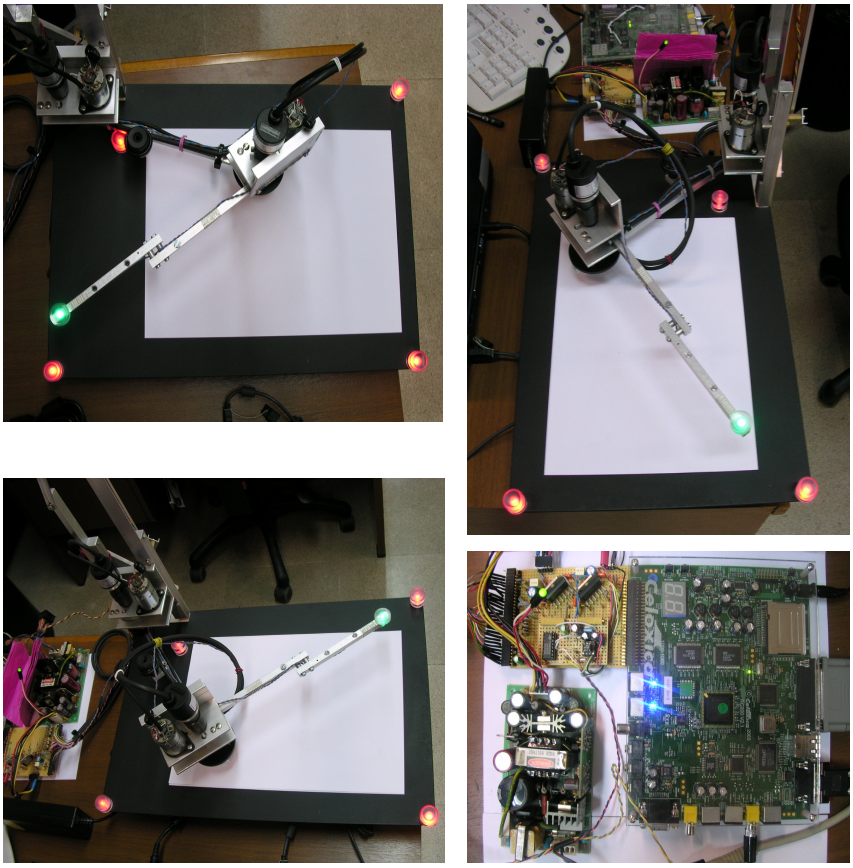

Figure 5: Bio-inspired robot arm.

The second platform is humanoid. With this platform we have studied the relationship between the human sensor systems (in particular the vestibular systems) and the industrial sensors performing the same task (maintenance balancing).

In this case we have described the controller based on actuador servo-systems emulating a real muscle. We have defined a neuron net for that behaves as a small GPG (General Patterns Generator). With this neuron net we have obtained a set of movements patterns that allows handling of the robots joins.

Different hypotheses are used in the field of the learning and how the real biological systems adapt themselves to changes in the environments conditions.

On the other hand, the interface between hardware robot and neuron simulators is presented. Finally, a vestibular sense response together with motor response during movements is presented to validate the system.

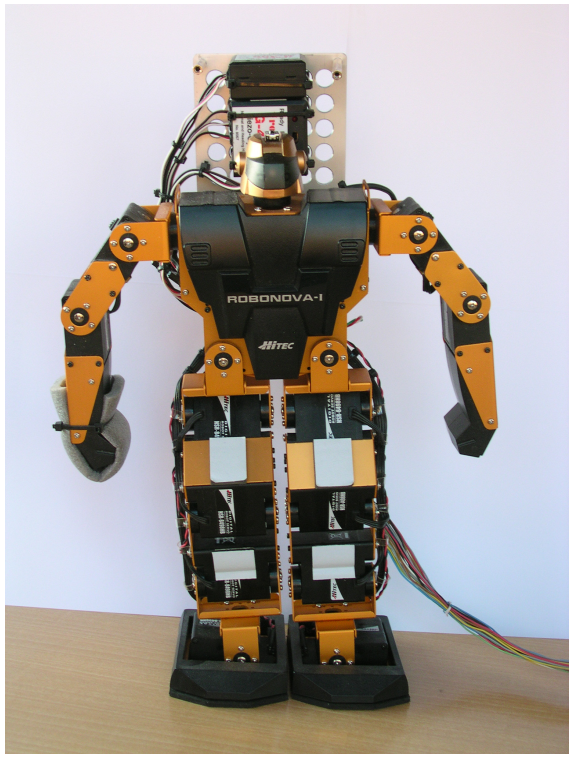

Figure 6: Adapted humanoid platform with a giro system.

5 Conclusions

The main contribution of this work consists in the development of two efficient processing architectures for real-time medium-scale spiking neural networks simulations (of biologically plausible neural systems). A time-driven approach has been developed with a hybrid Hardware/Software implementation. Here we have designed hardware computing modules as specific purpose hardware in FPGA and software modules that are run on a host conventional computer in which the hardware co-processing board is plugged in.

The other approach driven by events is radically different. In this case all the system is simulated in the co-processing board and the host computer is only used as initialization and monitorization tool. This second approach allows embedding the simulation engine in real robots without requiring the connection to conventional computers. Therefore it represents a stand-along platform.

The concepts that are studied, with this kind of simulation engines, require real-time robot control technologies. For this purpose, the work here presented describes how two robotic platforms have been developed and complemented with the communication modules to interface in real time the simulation engines. This contribution is a good example of the work that needs to be faced to integrate the different robotic technologies (motors and sensors), high performance computing technologies (reconfigurable hardware and parallel processing) and bio-inspired control schemes.

The main results of this Thesis are not specific simulations, but rather the performance of the simulation engines developed and the validation of the robotic platforms presented (including the communication schemes).

The work presented in this PhD Thesis opens the door to future research issues related with biologically plausible systems such as the cerebellum as simulation goal.

6 Main original contributions

The main original contributions of this work are the following:

1. Based on biologically plausible neural models a reference model has been adopted. This constitutes a trade-off between biological plausibility, computation complexity and customization capability adjusting specific functions and parameters (passive membrane potential decay, gradual injection of charge, etc.).

2. Based on the previous model, a time-driven processing architecture for simulating spiking neurons has been developed. This has been done designing a segmented datapath in order to take full advantage of the inherent parallelism of the computing resources of the FPGA devices. The system data (mainly neural spikes) has been structured in diverse configurations based on on-chip embedded memory and extended SRAM memory chips on-board. This allows the architecture to be fully scalable and the performance can be multiplied replicating functional units on the same chip.

3. A hybrid Software/Hardware platform has been designed for the utilization of the time-driven simulation engine. The network topology and the learning are simulated in software while the neural state variables are updated through dedicated hardware in the co-processing board (developed processing architecture). The gain in performance provided by the use of the co-processing board has been evaluated simulating in real-time an artificial cerebellum.

4. A spiking neural processing architecture driven by events has been developed. In this case the whole system has been implemented in the FPGA device (supported by several external memory chips). Software modules have been developed for monitoring the simulations. The different risks in the segmented datapath have been debugged. The whole system has been implemented in the reconfigurable platform. This facilitates its use as embedded control system. Outstanding performance rates (with more than two million spikes per second) have been obtained.

5. Two robotic platforms have been designed and complemented with specific communication modules for the developed simulation engines. In these platforms the direct access in real-time to sensors and motors has been facilitated. These platforms constitute very useful validation tools for bio-inspired processing schemes in which the continuous (real-time) interaction of the agent with its environment is required.

6. Making an analogy between the muscle functions in motor tasks and the dynamics of systems based on servo motors, a conversion mechanism of signals has been implemented to allow the movement control of the biped robot through the simulation engine.

7 Future work

The future work in this field is very diverse. On one hand, the presented work constitutes a very valid tool for the simulation of biologically plausible neural systems. In this work, the simulation technology developed has been presented (specific purpose processing architectures) and as future work remains its utilization for further concrete simulations of systems such as the cerebellum, the inferior olive, etc. Besides, robotic platforms have been developed and complemented allowing the experimentation of schemes for abstracting models of sensorimotor primitives and control schemes based on closed-loop perception-action cycles using real agents ("Embodiment"). This gives this technology a high potential in the robotic field and the exploration of bio-inspired control schemes with active agents.

8 References

[1] A. Delorme, S. Thorpe, "SpikeNET: An event-driven simulation package for modelling large networks of spiking neurons," Network: Computation in Neural Systems, vol. 14, pp.613-627, 2003.

[2] A. Janke, U. Roth, H. Klar, "A SIMD/dataflow architecture for a neurocomputer for spike-processing neural networks (NESPINN)," Proc. MicroNeuro'96, pp. 232-237, 1996.

[3] Celoxica, 2001-2004. [Online]. Available: http://www.celoxica.com

[4] Delorme, A., Gautrais, J. van Rullen, R., Thorpe, S. SpikeNET: A simulator for modelling large networks of integrate and fire neurons. Neurocomputing, Vols. 26-27, pp. 989-996. 1999.

[5] Delorme, A., Thorpe, S. SpikeNET: An event-driven simulation package for modelling large networks of spiking neurons. Network: Computation in Neural Systems, Vol. 14, pp. 613-627. 2003.

[6] E. Ros, R. Carrillo, E. M. Ortigosa, B. Barbour, R. Agís, "Event-driven simulation scheme for neural network models based on characterization look-up tables. Submitted to Neural Computation, 2005.

[7] G. Hartmann, G. Frank, M. Schaefer, C. Wolff, "SPIKE128K-An Accelerator for Dynamic Simulation of Large Pulse-Coded Networks," MicroNeuro'97, pp. 130-139, 1997.

[8] Glackin B., McGinnity T.M., Maguire L.P., Wu Q.X., Belatreche A., "A Novel Approach for the Implementation of Large Scale Spiking Neural Networks on FPGA Hardware", LNCS, pp. 552-563, 2005.

[9] L. Steels. “Emergent Functionality in Robotic Agents through On-Line Evolution”, in Brooks R.A., Maes, P.(eds.): Artificial Life IV, Proc. of the Fourth Int. Workshop on the Synthesis and Simulation of Living.

[10] M. Arnold, “Feedback learning in the olivary-cerebellar system,” PhD Thesis, The University of Sydney, 2001.

[11] M. Shaefer, T. Schoenauer, C. Wolff, G. Hartmann, H. Klar, U. Rueckert, “Simulation of Spiking Neural Networks – architectures and implementations”, Neurocomputing, vol. 48, 647-679, 2002.

[12] Manwani, A. and Koch, C. (1999). “Detecting and estimating signals in noisy cable structures“, I: Neuronal noise sources. Neural Comput., 11: pp.: 1797-1829.

[13] N. Mehrtash, D. Jung, H.H. Hellmich, T. Schoenauer, V.T. Lu, H. Klar, “Synaptic Plasticity in Spiking Neural Networks (SP^2INN): A System Approach,” IEEE Trans.Neural Networks, Vol. 14(5), 2003.

[14] N. Mehrtash, D. Jung, H.H. Hellmich, T. Schoenauer, V.T. Lu, H. Klar, “Synaptic Plasticity in Spiking Neural Networks (SP2INN): A System Approach,” IEEE Transactions on Neural Networks, vol. 14(5), 2003.

[15] R. Eckhorn, H.J. Reitboeck, M. Arndt, and P. Dicke, “Feature linking via stimulus evoked oscillations: Experimental results from cat visual cortex and functional implication from a network model,” in Proc. ICNN I, pp. 723–720, 1989.

[16] R. Eckhorn, R. Bauer, W. Jordan, M. Brosh, W. Kruse, M. Munk, H.J. Reitboeck, “Coherent oscillations: A mechanism of feature linking in the visual cortex?,” Biol. Cyber. Vol. 60, pp. 121-130, 1988.

[17] R. J. Vogelstein, U. Mallik and G. Cauwenberghs, “Beyond event-driven communication: dynamically-reconfigurable spiking neural systems”, The Neuromorphic Engineer, vol. 1(1), pp. 1,9, 2004.

[18] Ros, E., Carrillo, R., Ortigosa, E. M., Barbour, B., Agís, R., 2006 Event-driven Simulation Scheme for Spiking Neural Models based on Characterization Look-up Tables. Neural Computation, Vol. 18(12), pp. 2959-2993, 2006.

[19] Systems, MIT Press, 1994 L. Steels y P. Vogt, "Grounding adaptative language games in robotic agents", European Conference on Artificial Life, 1997.

[20] T. Schoenauer, S. Atasoy, N. Mehrtash, H. Klar, "NeuroPipe-Chip: A Digital Neuro-Processor for Spiking Neural Networks," IEEE Trans. Neural Networks, Vol. 13(1), pp. 205-213, 2002.

[21] T. Schoenauer, S. Atasoy, N. Mehrtash, H. Klar, "NeuroPipe-Chip: A Digital Neuro-Processor for Spiking Neural Networks," IEEE Trans. Neural Networks, vol. 13(1), pp. 205-213, 2002.

[22] Technical Library, Celoxica. [Online].

[23] V. Dante, P. Del Giudice and A. M. Whatley, "Hardware and software interfacing to address-event based neuromorphic systems", The Neuromorphic Engineer, vol. 2(1), pp. 5-6, 2005.

[24] W. Gerstner, W., Kistler, "Spiking Neuron Models," Cambridge University Press, 2002.

[25] White, J. A., Rubinstein, J. T., and Kay, A. R. (2000). "Channel noise in neurons". Trends Neuroscience. 23: pp.: 131-137.

[26] Xilinx, 1994-2003. [Online]. Available: http://www.xilinx.com.


ATC
ugr
Universidad
de Granada